Oxford Cambridge and RSA

An OCR endorsed
teaching and learning tool

OCR
A Level
Computer Science

H446 – Paper 1

5

Bitwise manipulation and masks

Unit 6
Data types

PG ONLINE

Objectives

- Perform logical, arithmetic and circular shifts on binary data
- Perform bitwise operations AND, OR and XOR
- Use masks to manipulate bits

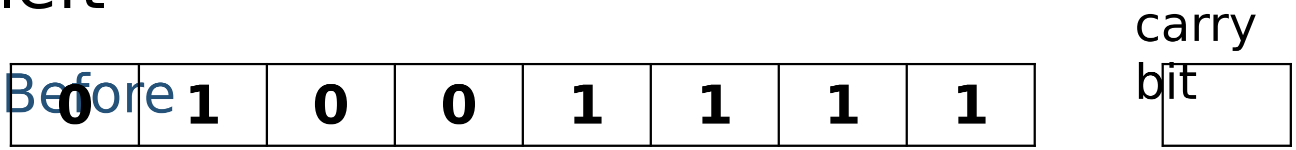
Manipulating bits

- The ability to manipulate individual bits in a byte is one of the advantages of assembly language
- Bits can be manipulated in two ways:
    - Using shift instructions to move bits left or right
    - Using logical instructions NOT, AND, OR and XOR with appropriate masks

0	0	0	1	1	0	1	0

Logical shift right

- A logical shift right causes the least significant bit (lsb) to be shifted into the carry bit, and a zero moves in to occupy the vacated space on the left

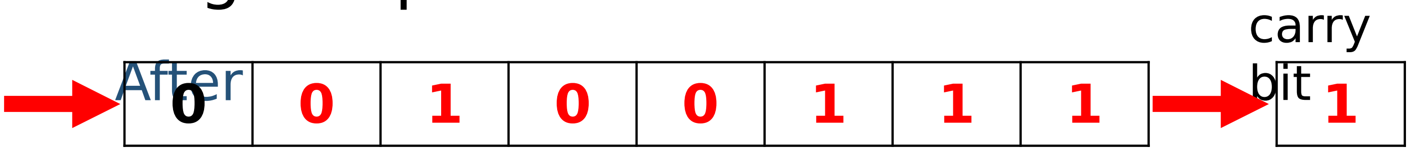


Before

0	1	0	0	1	1	1	1

carry bit

- Shift right 1 place



After

0	0	1	0	0	1	1	1

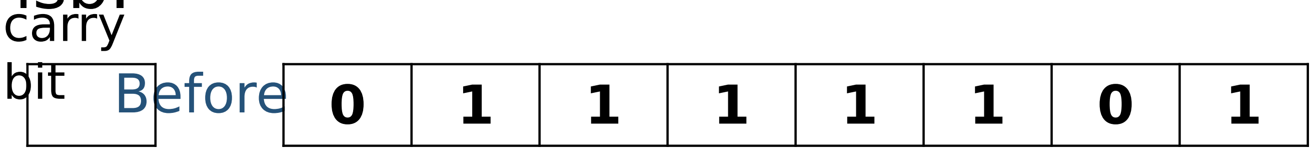
carry bit

1

- What is the result after shifting right 2 more places?

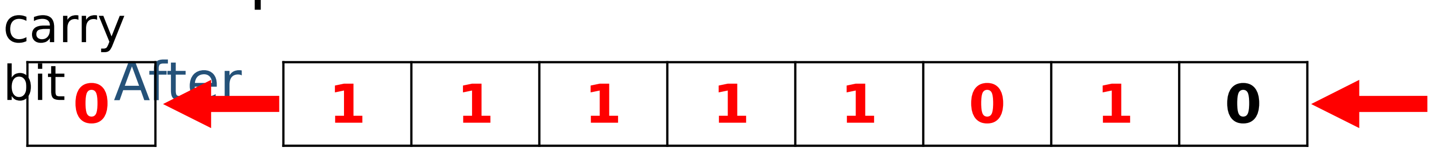
Logical shift left

- A logical shift left causes the most significant bit (msb) to be shifted into the carry bit, and a zero moves in to occupy the vacated space in the lsb.



carry bit	Before								
		0	**1**	**1**	**1**	**1**	**1**	**0**	**1**

- Shift left 1 place



carry bit	After								
0		**1**	**1**	**1**	**1**	**1**	**0**	**1**	**0**

- What is the result after shifting left 1 more place?

Use of a logical shift

- A logical right shift can be used to examine the least significant bit of a number.
- After the operation, the carry bit can be tested and a conditional branch executed
- Shifting left has the effect of multiplying a positive digit by 2
  - However, it won’t work with a two’s complement negative number – why not?

1	0	0	1	1	0	1	0

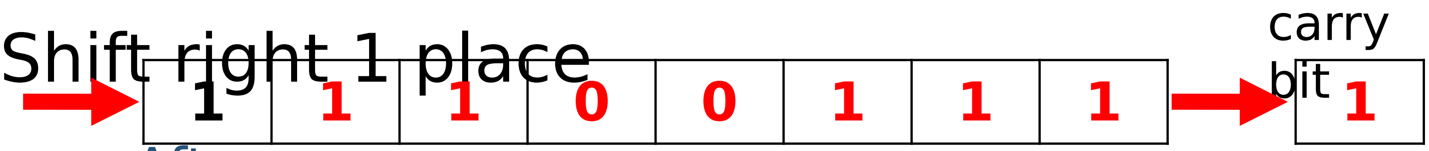
Arithmetic shift right

- An arithmetic shift right causes the least significant bit (lsb) to be shifted into the carry bit
- Instead of a zero moving in to occupy the vacated space, it pads with the msb to preserve the sign bit

									carry bit
1	**1**	**0**	**0**	**1**	**1**	**1**	**1**		

Before

- Shift right 1 place

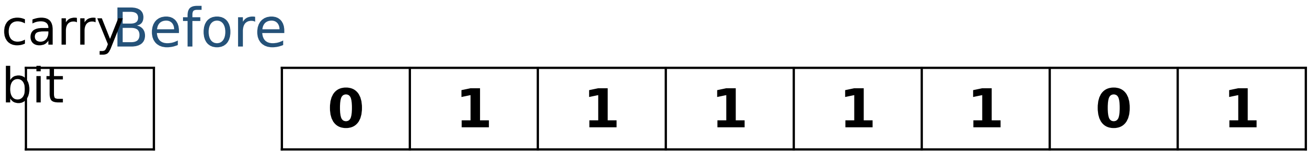


									carry bit
1	**1**	**1**	**0**	**0**	**1**	**1**	**1**		**1**

After

Arithmetic shift left

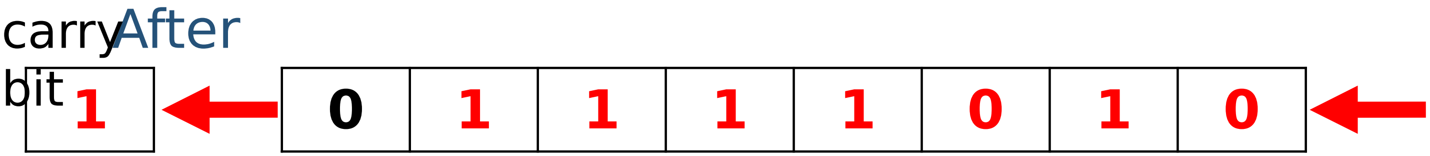
- An arithmetic shift left preserves the sign bit, shifting the second bit into the carry bit
- A zero moves in on the right to occupy the vacated space



Before

carry bit								
	0	1	1	1	1	1	0	1

- Shift left 1 place



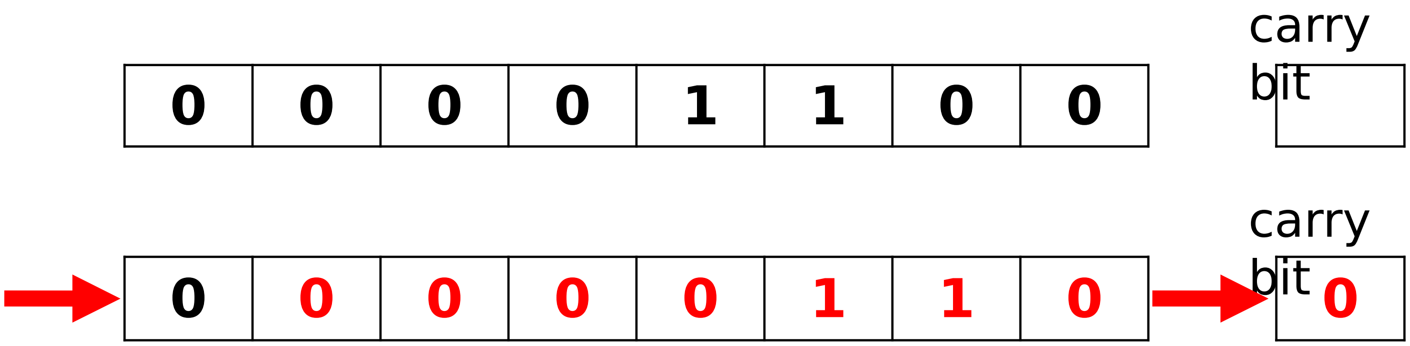
After

carry bit								
1	0	1	1	1	1	0	1	0

- What is the result after shifting left 1 more place?

Multiplication using arithmetic shifts

- Arithmetic shifts can be used to multiply and divide by multiples of 2
- What is the denary value before and after an arithmetic shift right one place?



								carry bit
0	0	0	0	1	1	0	0	

								carry bit
0	0	0	0	0	1	1	0	0

- What is the result after shifting right one more place? Two more places?

Multiplying two numbers

- Using a combination of shifts and addition, two binary numbers can be multiplied together
- Example: Multiply 17 by 5 using shifts and addition:

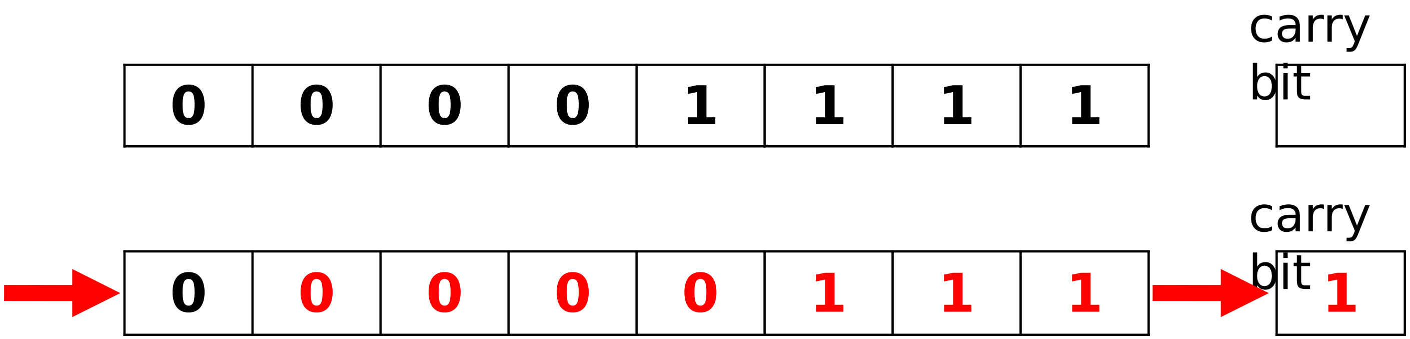
Multiply 17 by 1   0001 0001

Multiply 17 by 4 with 2 left shifts   0100 0100

Add together 0101 0101 = 85

Division using arithmetic shifts

- Note that if you divide an odd number by 2, it is rounded down
- What is the denary value before and after an arithmetic shift right one place?



								carry bit
0	0	0	0	1	1	1	1	
0	0	0	0	0	1	1	1	1

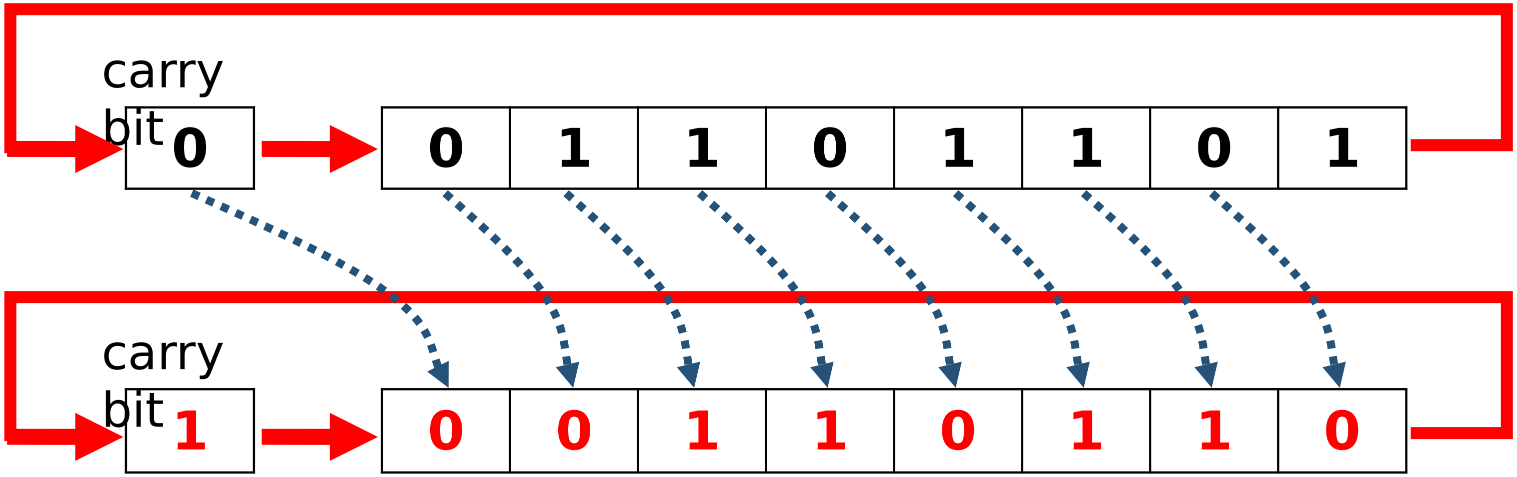
- If you shift right one place and then left one place, you don’t always end up with the value you started with!

Worksheet 5

- Now try questions 1 and 2 in **Task 1**

Circular shift right with carry

- With a circular shift right, the carry bit is moved into the msb and the value in the lsb is moved into the carry bit



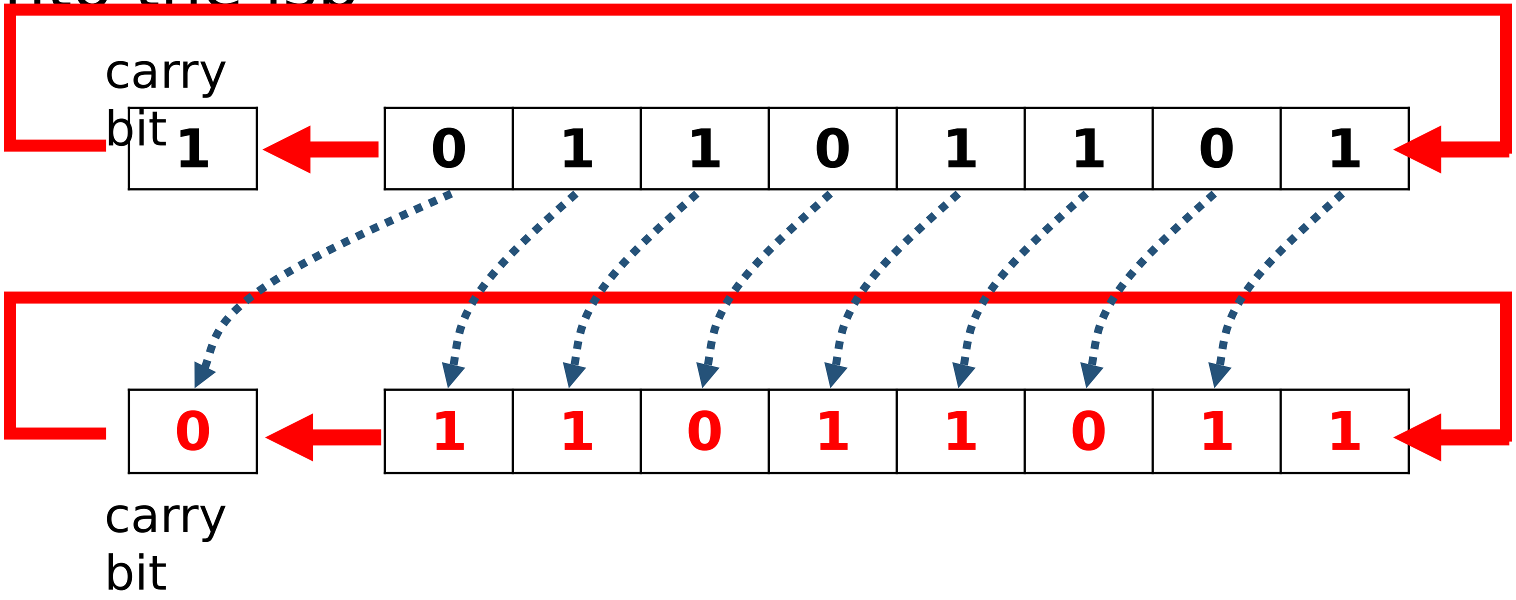
carry bit								
0	0	1	1	0	1	1	0	1

carry bit								
1	0	0	1	1	0	1	1	0

- Shift right 1 place

Circular shift left with carry

- With a circular shift left, the value in the msb is moved into the carry bit and the carry bit into the lsb



- Shift left 1 place

Worksheet 5

- Now try question 3 in **Task 1**

Logical instructions

- You have probably written programs involving complex Boolean conditions
- For example:
  - If (age >= 17) **and** (hasLicence = True) then …
  - If (temperature > 80) **or** (pressure > 65) then…
- In the first example, if condition A is true AND condition B is true, the complex condition is true
- In the second example, if A OR B, or both, are true, the complex condition is true

XOR

- There is another condition called XOR (exclusive OR), which is true if either of A or B, but not both, are true

Truth tables

AND OR XOR

A	B	OUT
False	False	False
False	True	
True	False	
True	True	

A	B	OUT
False	False	
False	True	
True	False	
True	True	

A	B	OUT
False	False	
False	True	
True	False	
True	True	

- Can you fill in the output columns for each of these tables?

Truth tables

AND OR XOR

A	B	OUT
False	False	False
False	True	False
True	False	False
True	True	True

A	B	OUT
False	False	
False	True	
True	False	
True	True	

A	B	OUT
False	False	
False	True	
True	False	
True	True	

Truth tables

AND OR XOR

A	B	OUT
False	False	False
False	True	False
True	False	False
True	True	True

A	B	OUT
False	False	False
False	True	True
True	False	True
True	True	True

A	B	OUT
False	False	
False	True	
True	False	
True	True	

Truth tables

AND OR XOR

A	B	OUT
False	False	False
False	True	False
True	False	False
True	True	True

A	B	OUT
False	False	False
False	True	True
True	False	True
True	True	True

A	B	OUT
False	False	False
False	True	True
True	False	True
True	True	False

Boolean logic

- In Boolean logic, 1 represents True and 0 represents False
- True AND True = True, i.e. 1 AND 1 = 1
- True AND False = False , i.e. 1 AND 0 = 0
- True OR True = True, i.e. 1 OR 1 = ?
- True OR False = ? i.e. 1 OR 0 = ?
- True XOR False = ? i.e. 1 XOR 0 = ?

Boolean logic

- The instructions AND, OR and XOR can be summarised in the table below:

	AND	OR	XOR
Input A	1010	1010	1010
Input B	1100	1100	1100
Result	?	?	?

Boolean logic

- The instructions AND, OR and XOR can be summarised in the table below:

	AND	OR	XOR
Input A	1010	1010	1010
Input B	1100	1100	1100
Result	1000	?	?

Boolean logic

- The instructions AND, OR and XOR can be summarised in the table below:

	AND	OR	XOR
Input A	1010	1010	1010
Input B	1100	1100	1100
Result	1000	1110	?

Masks with Boolean logic

- The instructions AND, OR and XOR can be summarised in the table below:

	AND	OR	XOR
Input A	1010	1010	1010
Input B	1100	1100	1100
Result	1000	1110	0110

- Input B is a mask, which in combination with the Boolean operator, will set, clear or toggle the input bits

Application of AND

- An AND operator may be used to clear a particular bit, leaving the other bits unchanged
- Suppose you want to clear the rightmost 4 bits
- What mask will you use?

AND

Input A	1010 1101
Input B	?
Result	1010 0000

Application of AND

- An AND operator may be used to clear a particular bit, leaving the other bits unchanged
- Suppose you want to clear the rightmost 4 bits
- Input B is the mask, AND is the logical operator

AND

Input A	1010 1101
Input B	1111 0000
Result	1010 0000

Application of OR

- An OR operator may be used to set a particular bit, leaving the other bits unchanged
- Suppose you want to set bits 1, 2, 7, 8 (leftmost bit is bit 1) leaving the others unchanged
- What mask will you use?

OR

Input A	1010 1101
Input B	?
Result	1110 1111

Application of OR

- An OR operator may be used to set a particular bit, leaving the other bits unchanged
- Suppose you want to set bits 1, 2, 7, 8 (leftmost bit is bit 1) leaving the others unchanged
- Input B is the mask, OR is the logical operator

OR	
Input A	1010 1101
Input B	1100 0011
Result	1110 1111

Application of XOR

- An XOR operator may be used to toggle a particular bit, leaving the other bits unchanged
- Toggle bits 1-4, leaving 5-8 unchanged
- What mask will you use?

XOR

Input A	1010 1101
Input B	?
Result	0101 1101

Application of XOR

- An XOR operator may be used to toggle a particular bit, leaving the other bits unchanged
- Toggle bits 1-4, leaving 5-8 unchanged
- Input B is the mask, XOR is the logical operator

XOR

Input A	1010 1101
Input B	1111 0000
Result	0101 1101

Worksheet 5

- Try the questions in **Task 2** of the worksheet

Summary of masks and operators

- To **clear** a particular bit, use a mask of **0** and the **AND** operator. Use a mask of **1** to leave a bit unchanged
- To **set** a particular bit, use a mask of **1** and the **OR** operator. Use a mask of **0** to leave a bit unchanged
- To **toggle** a particular bit, use a mask of **1** and the **XOR** operator. Use a mask of **0** to leave a bit unchanged

Plenary

- There are three different types of shift instruction:
  - Logical, arithmetic and circular
- Arithmetic shifts are commonly used by computers to perform multiplication
- The Boolean operators AND, OR and XOR may be used in combination with masks to manipulate particular bits in a byte

Copyright

Licence agreement

PG ONLINE